

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日            2 0 0 3 年   2 月 1 8 日  
Date of Application:

出 願 番 号            特 願 2 0 0 3 - 0 4 0 0 5 6  
Application Number:

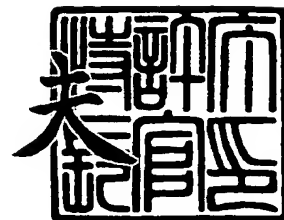
[ST. 10/C]:            [ J P 2 0 0 3 - 0 4 0 0 5 6 ]

出      願      人            株 式 会 社 デ ン ソ ー  
Applicant(s):

2 0 0 3 年 1 2 月 1 9 日

特許庁長官  
Commissioner,  
Japan Patent Office

今 井 康



【書類名】 特許願

【整理番号】 PNID4203

【提出日】 平成15年 2月18日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/46

【発明者】

    【住所又は居所】 愛知県刈谷市昭和町 1 丁目 1 番地 株式会社デンソー内

    【氏名】 西村 忠治

【特許出願人】

    【識別番号】 000004260

    【氏名又は名称】 株式会社デンソー

【代理人】

    【識別番号】 100082500

    【弁理士】

    【氏名又は名称】 足立 勉

    【電話番号】 052-231-7835

【手数料の表示】

    【予納台帳番号】 007102

    【納付金額】 21,000円

【提出物件の目録】

    【物件名】 明細書 1

    【物件名】 図面 1

    【物件名】 要約書 1

    【包括委任状番号】 9004766

【プルーフの要否】 要

## 【書類名】明細書

【発明の名称】タスク間通信方法、プログラム、記録媒体、電子機器

## 【特許請求の範囲】

## 【請求項 1】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理における前記受信側タスクの起床要求の回数を前記データの送信回数よりも少ない回数とし、

前記受信側タスク上での処理における前記キューに格納されたデータを取得する処理において、前記キューに格納されたデータを複数取得すること

を特徴とするタスク間通信方法。

## 【請求項 2】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理で、前記受信側タスク上での処理に対するデータ

の送信要求が発生した場合に、当該受信側タスク上での処理から取得可能な前記キュー内に、前記キューへの当該データの格納以前にすでに格納されているデータがあるか否かを判定し、ある場合には前記送信側タスク上での処理における当該受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行い、

前記受信側タスクでは、当該受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得すること

を特徴とするタスク間通信方法。

### 【請求項 3】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理で、前記受信側タスク上での処理に対するデータの送信要求が発生した場合に、当該受信側タスクに対する前記起床要求が現在あるか否かを判定し、ある場合には前記送信側タスク上での処理における当該受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行い、

前記受信側タスクでは、当該受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得すること

を特徴とするタスク間通信方法。

### 【請求項 4】

請求項 1～3 のいずれかに記載のタスク間通信方法において、

前記オペレーティングシステムは、前記起床要求があつて未だ実行していないタスクを特定するためのタスク特定情報を記憶し、記憶されたタスク特定情報に

対応する優先度が一番高いタスクから順に、タスクを起床させるものであり、すべてのタスクを異なる優先度に設定し、前記キューは、当該優先度ごとに設けること

を特徴とするタスク間通信方法。

**【請求項 5】**

請求項 1 ～ 4 のいずれかに記載のタスク間通信方法の各処理をコンピュータに実行させるためのプログラム。

**【請求項 6】**

請求項 5 に記載のプログラムを記録した記録媒体。

**【請求項 7】**

請求項 5 に記載のプログラムと、当該プログラムを実行するコンピュータとを備えることを特徴とする電子機器。

**【発明の詳細な説明】**

**【 0 0 0 1 】**

**【発明の属する技術分野】**

タスク間通信方法等に関する。

**【 0 0 0 2 】**

**【従来の技術】**

従来からリアルタイムオペレーティングシステム（R T O S）等によって管理されるタスク間で、データを含むメッセージの通信（送受信、やりとり、受け渡し）を行うための種々のタスク間通信方法が知られている（例えば、特許文献 1 参照。）。

**【 0 0 0 3 】**

例えば、図 4（a）は、メッセージを送信する側のタスク内（あるいはそのタスクから呼び出すサブルーチン内）におけるメッセージ送信処理の流れを示すフローチャートである。また図 4（b）は、メッセージ内のデータを受信（取得）する側のタスク（あるいはそのタスクから呼び出すサブルーチン内）におけるデータ受信利用処理の流れを示すフローチャートである。

**【 0 0 0 4 】**

ここで、メッセージの送信側タスクをタスクA、メッセージに基づいてRTOSによって起床されメッセージ内のデータを受信して利用する受信側タスクをタスクBとして説明する。タスクAは優先度Aであり、タスクBは優先度Bであり、優先度Aは優先度Bよりも優先度が高いものとする。RTOSは、各タスクから起床待ちキュー（Ready Queue）にキューイングされたタスクのうち、優先度の最も高いタスクを起床させるスケジューリング方法を採用するものとする。

#### 【0005】

例えば、タスクA内でイベントの発生を検知した場合に、そのイベントに関連するデータを、タスクBのそのイベント用の処理ルーチンで処理させるためにメッセージ通信処理が行われる。

タスクAにおけるメッセージ送信処理は、図4（a）に示すように、送信対象のデータをメッセージ送信先のタスク用のキュー、すなわち、タスクB用のキュー（Queue）にキューイング（保存）し（S11）、RTOSに対し、対応する送信先タスク（受信側タスク）、すなわちタスクBの起床要求を行う（S12）処理である。

#### 【0006】

RTOSは、この起床要求を受け、この起床要求されたタスクを特定するための識別情報（タスクID）をタスクの起床待ちキューにキューイングする。そして、優先度AのタスクAの処理が完了すると、RTOSはタスクBの優先度よりも高い優先度のタスクのタスクIDが起床待ちキューになくなった時点で、起床待ちキューにキューイングされているタスクIDに基づいてタスクBを起床し、タスクBを実行する。

#### 【0007】

このタスクB内では、図4（b）に示す処理が行われる。つまり、自タスク用のキュー、すなわちタスクBのキューからデータを取り出し（S21）、そのデータを用いた処理を行う（S22）。このようにして、タスクA内の処理からタスクBを起床させてタスクB内の処理を実行させ、このタスクB内の処理で、タスクA内の処理によってキューイングされたデータを受け取って利用することが

できる。例えば、タスク A 内で検知したイベントの処理に必要なデータを、タスク B 内のこのイベントの処理ルーチンに渡して処理させることができる。

【0008】

【特許文献 1】

特開 2002-189606 号公報

【0009】

【発明が解決しようとする課題】

こうした従来のタスク間通信方法では、優先度が高いタスク A 内で、優先度の低いタスク B に対してメッセージを一度に複数回送信する場合、図 4 (a) に示したメッセージ送信処理は、複数回実行されることとなる。

【0010】

したがって、メッセージを送信する回数分のデータが、一旦、送信先のタスク B 用のキューにキューイングされるとともに、タスク B の起床要求が R T O S のタスクの起床待ちキュー ( R e a d y   Q u e u e ) にキューイングされる。

そして、その後、タスク B よりも優先度の高い処理 (タスク、割り込み) が無くなった時点で、1. R T O S によるタスク B の起床処理 → 2. タスク B 内のデータ受信処理 ( S 2 1 ) → 3. タスク B における受信したデータを用いた処理 ( S 2 2 ) → 4. タスク B 終了という一連のメッセージ配送処理を、その複数回分繰り返し実行する必要がある。

【0011】

例えば、図 5 は、図 4 (a) の処理を行うルーチンを SendMessage 関数、図 4 (b) の S 2 1 の処理を行うルーチンを ReceiveMessage 関数としてサブルーチン化して持つイベントサービスを介して、タスク A からタスク B 内のイベントの処理ルーチンにデータ A、データ B、データ C に基づく処理を実行させるための 3 つのメッセージを送る場合のタスク間メッセージ通信の処理の流れを示す説明図である。なお、イベントサービスは R T O S の一部として構成することもでき、タスクの一部 (サブルーチン) として構成することもできる。ここでは、イベントサービスによる処理は、イベントサービスの関数の呼び出し元と同じ実行レベル (優先度) で実行されるものとして説明する。

## 【0012】

図5に示すように、タスクA内でイベントサービスのSendMessage関数を呼び出し、データAをその引数としてイベントサービスへ渡す。イベントサービスは、この関数内でタスクB用のキューにデータAをキューイングし（図4（a）のS11に相当する）、RTOSに対してタスクBの起床要求をする（図4（a）のS12に相当する）。そしてこの起床要求に基づきRTOSは、起床待ちキューにタスクBのタスクIDをキューイングする。そして、再びタスクAは、SendMessage関数を呼ぶことでデータBをイベントサービスへ渡す。イベントサービスはタスクB用のキューにデータBをキューイングし（S11）、RTOSに対してタスクBの起床要求をする（S12）。この要求に基づきRTOSは、起床待ちキューにタスクBのタスクIDをキューイングする。そして、再びタスクAは、SendMessage関数を呼ぶことでデータCをイベントサービスへ渡す。イベントサービスはタスクBのキューにデータCをキューイングし（S11）、RTOSに対してタスクBの起床要求をする（S12）。この要求に基づきRTOSは、起床待ちキューにタスクBのタスクIDをキューイングする。そして、タスクAのすべての処理が終了すると、RTOSへ処理が移行する。

## 【0013】

RTOSは、起床待ちキューを参照して、このキュー内で優先度の最も高いタスクIDに対応するタスクを起床させる。起床待ちキューには、タスクBのタスクIDがキューイングされているので、タスクBを起床させる（図5における一番上のSchedule）。ここで起床待ちキューに残されたタスクBのタスクIDは残り2つとなる。起床されたタスクB内の処理では、イベントサービスのReceiveMessage関数を呼ぶ。イベントサービスは、ReceiveMessage関数の処理として、タスクB用にキューイングされているデータAを取り出し（図4のS21に相当する）、タスクBにそのデータAを返す。そしてタスクBは、イベントサービスから受け取ったデータAを用いた処理を実行する（図4のS22に相当する）。そしてタスクBのその他の処理が終了すると、処理をRTOSへ移行させる。RTOSは、起床待ちキューを参照して、優先度の高い順にタスクを起床させる。起床待ちキューには、タスクBがキューイングされているので、タスクBを起床さ



せる。ここで起床待ちキューに残されたタスク B のタスク ID は残り 1 つとなる。起床されたタスク B 内の処理では、イベントサービスの ReceiveMessage 関数を呼ぶ。イベントサービスは、ReceiveMessage 関数の処理として、タスク B 用にキューイングされているデータ B を取り出し (S 21)、タスク B にそのデータ B を返す。そしてタスク B は、イベントサービスから受け取ったデータ B を用いた処理を実行する (S 22)。そしてタスク B のその他の処理が終了すると、処理を R T O S へ移行させる。R T O S は、起床待ちキューを参照して、優先度の高い順にタスクを起床させる。起床待ちキューにはまだタスク B がキューイングされているので、タスク B を起床させる。ここで起床待ちキューは空になる。起床されたタスク B 内の処理では、イベントサービスの ReceiveMessage 関数を呼ぶ。イベントサービスは、ReceiveMessage 関数の処理として、タスク B 用にキューイングされているデータ C を取り出し (S 21)、タスク B にそのデータ C を返す。そしてタスク B は、イベントサービスから受け取ったデータ C を用いた処理を実行する (S 22)。そしてタスク B のその他の処理が終了すると、処理を R T O S へ移行させる。

#### 【0014】

このように、送信側タスク (タスク A) 内で処理を R T O S へ返すことなく、送信先のタスク (タスク B) にメッセージを複数回送信した場合に、受信側タスクであるタスク B は、単にタスク B 用のキュー内のデータを受信してそのデータを用いる処理を順次実行するだけにもかかわらず、1 メッセージ毎にタスクの起床・終了処理 (R T O S における処理 (例えば図 5 の schedule のための処理など) やタスク内での処理 (例えば図 4 (b) の処理へ移行するための判定処理など) が発生し、処理装置 (例えば C P U) の処理負荷が大きくなるという問題がある。

#### 【0015】

そこで本願発明は、処理装置の処理負荷を低減させることのできるタスク間通信方法等を提供することを目的とする。

#### 【0016】

【課題を解決するための手段及び発明の効果】

上述した問題点を解決するためになされた請求項1に記載のタスク間通信方法は、送信側タスク上での処理から受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、受信側タスク上での処理から取得可能なキューに格納し、その受信側タスクの起床要求をオペレーティングシステムに対して行う。このとき、送信側タスク上での処理における前記受信側タスクの起床要求の回数をデータの送信回数よりも少ない回数とする。そして、その起床要求に基づいてオペレーティングシステムの処理によって受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに格納されたデータを複数取得する処理を行う。

#### 【0017】

したがって、送信側タスクでは複数のデータの送信要求に対して、1の起床要求を行い、その起床要求によって起床された受信側タスクでは、複数のデータをキューから取得するため、従来のように1つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

#### 【0018】

また、上述した問題点を解決するためになされた請求項2に記載のタスク間通信方法は、送信側タスク上での処理から、受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、その受信側タスク上での処理から取得可能なキューに格納し、そのキュー内に、前記キューへのそのデータの格納以前にすでに格納されているデータがあるか否かを判定し、ある場合にはその受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行う。そして、起床要求に基づいて前記オペレーティングシステムの処理によってその受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得する。

#### 【0019】

このようにすれば、送信側タスク上の処理で、連続してデータ（例えばメッセージ）を受信側タスク上の処理へ送る場合に、その受信側タスクの起床要求は1回のみで済む。したがって、従来のように1つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

#### 【0020】

また、上述した問題点を解決するためになされた請求項3に記載のタスク間通信方法は、送信側タスク上での処理から、受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、その受信側タスク上での処理から取得可能なキューに格納し、その受信側タスクに対する起床要求が現在あるか否かを判定し、ある場合にはその受信側タスクに対する起床要求を行わず、ない場合にのみ起床要求を行う。そして、起床要求に基づいて前記オペレーティングシステムの処理によってその受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得する。

#### 【0021】

このようにすれば、送信側タスク上の処理で、連続してデータ（例えばメッセージ）を受信側タスク上の処理へ送る場合に、その受信側タスクの起床要求は1回のみで済む。したがって、従来のように1つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

#### 【0022】

特に、オペレーティングシステムがタスクの優先度に基づいて優先度の高いタスクから順に実行する構成の場合、請求項4に示すように、すべてのタスクを異なる優先度に設定し、前記キューはその優先度ごとに設けるようにするとよい。このようにすれば、優先度に基づいてタスクやキューを管理できる。すなわち、請求項1～3に記載の「受信側タスク上での処理から取得可能なキュー」は、優

先度ごとに設けたキューとすることができる。例えば、データ（メッセージ）送信時に、他に送信待ちの同一優先度のメッセージがキューに有るか否かをチェックし、有る場合にはオペレーティングシステムに対するその優先度のタスクの起床要求を行わず、ない場合のみその優先度のタスクの起床要求を行うようにできる。なお、このタスクの起床要求がすでにあるか否かは、例えば請求項2に示すようにその優先度のキューにデータがすでにあるか否かに基づいて判定してもよいし、請求項3に示すようにその受信側タスクに対する前記起床要求が現在あるか否かに基づいて判定してもよい。そして、受信側タスク上での処理では、データの受信処理時に、その優先度のキューにメッセージがあれば、そのメッセージをすべて取得するように構成できる。このように構成することで、例えば、送信側タスクで同一優先度のメッセージが複数連続して送信要求され、かつ、受信側タスクが送信側タスクに比べ優先度が同じか低い場合に、受信側タスクの起床及び終了処理を減らすことができ、処理装置の処理負荷が低減できる。

#### 【0023】

そして、請求項5に示すように、上述したタスク間通信方法の各処理をコンピュータに実行させるためのプログラムとして構成することができる。このようなプログラムは、例えば、請求項6に示すようにして、フレキシブルディスク、光磁気ディスク、CD-ROM、ハードディスク、ROM、RAM等のコンピュータ読み取り可能な記録媒体に記録し、必要に応じてコンピュータシステムにロードして起動することにより実行することができる。また、プログラムは、ネットワークを介してロードして起動することにより実行することもできる。また、請求項7に示すように、請求項5に記載のプログラムと、このプログラムを実行するコンピュータを備える電子機器として構成することができる。このような電子機器は、コンピュータ（処理装置）によるタスクの起動・終了処理を減らすことができるため、機器本来の機能を実現するための処理に処理装置の処理時間をより多く割くことが可能となる。

#### 【0024】

#### 【発明の実施の形態】

以下、本発明を具体化した一実施例について図面を参照して説明する。図1は

、本発明の電子機器としてのエンジン制御装置（以下「ECU」という。）1の構成を表すブロック図である。ECU1は、車両に搭載された内燃機関型エンジンの制御を行う。

【0025】

ECU1は、エンジンのクランク軸が所定角度回転する毎にパルス状の信号を出力する回転角センサ、エンジンの特定の気筒のピストンが所定位置（例えば上死点：TDC）にくる度にパルス状の信号を出力する基準位置センサ、エンジンの冷却水の温度を検出する水温センサ、及び酸素濃度を計測する酸素濃度センサ等、エンジンの運転状態を検出する様々なセンサ30からの信号を入力して波形整形やA/D変換を行ったり、車内LAN32から信号を入力したりする入力回路21と、入力回路21からの入力信号に基づき、エンジンを制御するための様々な処理や車内LAN32を介して他のECUからデータを受信したり他のECUに対して送信するデータを生成して送信したりするための様々な処理を実行するマイコン10と、マイコン10からのデータに応じて、エンジンに取付けられたインジェクタ（燃料噴射装置）及びイグナイタ（点火装置）等のアクチュエータ40を駆動したり、車内LAN32にデータを出力したりするための出力回路22とを備えている。

【0026】

そして、マイコン10には、プログラムを実行する周知のCPU11と、CPU11によって実行されるプログラムを記憶するROM12と、CPU11による演算結果等を記憶するためのRAM13と、入力回路21及び出力回路22との間で信号をやり取りするためのI/O14と、各種レジスタやフリーランカウンタ等（図示省略）とを備えている。

【0027】

このように構成されたECU1は、入力回路21及び出力回路22を制御し車内LAN32を介して他のECU等と通信を行う通信処理や、各種センサ30及び車内LAN32から入力回路21を介して入力される信号に基づき、出力回路22に接続されたアクチュエータ40を駆動するエンジン制御処理を行う。

【0028】

次に、ROM 12 に記憶される「プログラム」としてのエンジン制御プログラムの構成を説明する。なお、本明細書では、例えば「タスクが・・・する」「RTOS が・・・する」「イベントサービスが・・・する」「・・・関数が・・・する」というプログラムを主体とした表現を適宜用いているが、詳しくは、CPU 11 が ROM 12 に記憶されたこれらのプログラムを実行して処理を行うことは言うまでもない。エンジン制御プログラムは、RTOS、タスク A、タスク B、タスク c・・・等の各々のタスク、及び、各々のタスクのメッセージ送信処理のサブルーチンである SendMessage 関数とデータ受信処理のサブルーチンである ReceiveMessage 関数を備えるイベントサービスを構成するプログラムを備える。各タスクはそれぞれ異なる優先度に設定されており、イベントサービスの各関数（ルーチン）は呼び出し元のタスクと同じ優先度で実行される。

#### 【0029】

RAM 13 上には各タスク用のキューのための記憶領域を確保している。キューは、先入れ先出し（FIFO）バッファ（記憶領域）である。

例えば、タスク A 内の処理においてタスク B 内の所定のルーチンで処理すべきイベントが発生した場合のように、タスク A 内の処理からタスク B 内の処理へデータを送る必要が発生した場合、タスク A 内の処理で、送るべきデータを引数としてイベントサービスの SendMessage 関数を呼ぶ。

#### 【0030】

このイベントサービスの SendMessage 関数による処理を図 2（a）に示す。まず、SendMessage 関数による処理では、引数として受け取った（例えばスタックを介して受け取った）送信対象のデータを、送信先のタスク用のキュー、すなわちタスク B 用のキューへ、キューイングする（S110）。

#### 【0031】

そして、この送信先のタスク用のキューのメッセージ数カウンタの値、すなわちタスク B 用のキューのメッセージ数カウンタの値を、インクリメントする（S120）。なお、各タスク用のメッセージ数カウンタの値はRTOSの初期化処理で0に設定されている。

#### 【0032】

次に、この送信先のタスク用のメッセージ数カウンタの値、すなわちタスク B 用のメッセージ数カウンタの値が、1 であるか否かを判定する (S 130)。1 の場合には (S 130: YES)、S 140 へ移行する。S 140 では、OS に対して、送信先タスクの起床要求、すなわち、タスク B の起床要求を行う。RTOS は、この起床要求を受け、この起床要求されたタスクを特定するための識別情報 (タスク ID) をタスクの起床待ちキューにキューイングする。すなわち、タスク B のタスク ID を起床待ちキューにキューイングする。そして、この Send Message 関数による処理を終了し、タスク A の処理へ復帰する。一方、S 130 の判定で、メッセージカウンタの値が 1 以外の場合には (S 130: NO)、S 140 の処理は行わずに、そのまま、この SendMessage 関数による処理を終了し、タスク A の処理へ復帰する。

#### 【0033】

このように、送信先のタスク用のメッセージカウンタの値が 1 の場合、すなわち、送信先のタスク用のキューに最初に送信対象のデータを保存したときにだけ、その送信先のタスクの起床要求を OS に対して行い、それ以後、キューに送信対象のデータを追加する際には、OS に対する送信先タスクの起床要求を行わない。

#### 【0034】

イベントサービスの SendMessage 関数から復帰した後、タスク A はその他のタスク内の処理を行う。そして、タスク A の処理が終了するとタスク A は RTOS へ処理の終了を伝え (処理を RTOS へ移行して)、終了する。

RTOS は、起床待ちキューに保存されたタスク ID のうち、もっとも優先度の高いタスク ID のタスクを起床させる。すなわちそのタスクへ処理を移行させる。ここでは、データの受信側タスク、すなわちタスク B が起床された場合について説明する。

#### 【0035】

このタスク B 内 (データ受信タスク内) では、図 2 (b) に示すデータ受信利用処理を行う。まず、自タスク用のキュー、つまりタスク B 用のキューからデータを取り出す (S 210)。この S 210 の処理はイベントサービスの ReceiveM

essage関数をタスクB内から呼び出し、イベントサービスのReceiveMessage関数の処理で、タスクB用のキューからデータを取り出してタスクBへ渡すことを行う。そして、タスクBでは、このデータを用いた処理を行う（S220）。続いて、このタスク用のメッセージカウンタ、すなわちタスクB用のメッセージカウンタの値をデクリメントし（S230）、自タスク用のメッセージカウンタの値、すなわちタスクB用のメッセージカウンタの値が0であるか否かを判定する（S240）。0でない場合には（S230：NO）、S210へ移行して、再び、S210～S240の処理を行う。一方、0の場合には、このデータ受信利用処理を終了する。そしてタスクBのその他の処理を行い、すべての処理が終了すると、タスクBはRTOSへ処理の終了を伝え（処理をRTOSへ移行して）、終了する。

#### 【0036】

このようにして、送信側タスク内の処理（タスクA内の処理）から受信側タスク（タスクB）を起床させて、受信側タスク内の処理（タスクB内の処理）を実行させ、この受信側タスク内の処理で、送信側タスク内の処理（タスクA内の処理）によってキューイングされたデータを受け取って利用することができる。例えば、タスクA内で検知したイベントの処理に必要なデータを、タスクB内のこのイベントの処理ルーチンに渡して処理させることができる。

#### 【0037】

ここで、タスクAからタスクBへメッセージを3回連続して発行（送信）する場合の処理の例を図3に示して説明する。

図3に示すように、タスクA内でイベントサービスのSendMessage関数を呼び出し、データAをその引数としてイベントサービスへ渡す。イベントサービスは、この関数内でタスクB用のキューにデータAをキューイングし（図2（a）のS110に相当する）、タスクB用のメッセージ数カウンタの値をインクリメントする（S120）。その結果、メッセージ数カウンタの値は1となるので（S130：YES）、RTOSに対してタスクBの起床要求を行う（S140）。そしてこの起床要求に基づきRTOSは、起床待ちキューにタスクBのタスクIDをキューイングする。そして、再びタスクAは、SendMessage関数を呼ぶこと



でデータ B をイベントサービスへ渡す。イベントサービスはタスク B 用のキューにデータ B をキューイングし (S 1 1 0)、タスク B 用のメッセージカウンタの値をインクリメントする。その結果、メッセージ数カウンタの値は 2 となるので、そのままタスク A の処理へ、復帰する (S 1 3 0 : NO)。すなわち図 3 に示すように、2 つ目のデータであるデータ B をキューに入れた際には、タスク B の起床要求は行わない。そして、再びタスク A は、SendMessage 関数を呼ぶことでデータ C をイベントサービスへ渡す。イベントサービスはタスク B のキューにデータ C をキューイングし (S 1 1 0)、タスク B 用のメッセージカウンタの値をインクリメントする。その結果、メッセージ数カウンタの値は 3 となるので、そのままタスク A の処理へ、復帰する (S 1 3 0 : NO)。すなわち図 3 に示すように、3 つ目のデータであるデータ C をキューに入れた際には、タスク B の起床要求は行わない。そして、タスク A のすべての処理が終了すると、RTOS へ処理を移行する。

#### 【0038】

RTOS は、起床待ちキューを参照して、このキュー内で優先度の最も高いタスク ID に対応するタスクを起床させる。起床待ちキューには、タスク B のタスク ID がキューイングされているので、タスク B を起床させる (図 3 における Schedule)。ここで起床待ちキューは空になる。起床されたタスク B 内の処理では、イベントサービスの ReceiveMessage 関数を呼ぶ。イベントサービスは、ReceiveMessage 関数の処理として、タスク B 用にキューイングされているデータ A を取り出し (図 4 の S 2 1 0 に相当する)、タスク B にそのデータ A を返す。そしてタスク B は、イベントサービスから受け取ったデータ A を用いた処理を実行する (S 2 2 0)。そして、タスク B 用のメッセージ数カウンタの値をデクリメントする (S 2 3 0)。その結果、タスク B 用のメッセージ数カウンタの値は、2 となるので、S 2 4 0 の判定処理で再び S 2 1 0 の処理へ移行することとなる。再び、イベントサービスの ReceiveMessage 関数を呼び、イベントサービスは、ReceiveMessage 関数の処理として、タスク B 用にキューイングされているデータ B を取り出して (S 2 1 0)、タスク B にそのデータ B を返す。そしてタスク B は、イベントサービスから受け取ったデータ B を用いた処理を実行する (S 2 2 0)。

。そして、タスク B 用のメッセージ数カウンタの値をデクリメントする（S 2 3 0）。その結果、タスク B 用のメッセージ数カウンタの値は、1 となるので、S 2 4 0 の判定処理で再び S 2 1 0 の処理へ移行することとなる。再び、イベントサービスの ReceiveMessage 関数を呼び、イベントサービスは、ReceiveMessage 関数の処理として、タスク B 用にキューイングされているデータ C を取り出して（S 2 1 0）、タスク B にそのデータ C を返す。タスク B は、イベントサービスから受け取ったデータ C を用いた処理を実行する（S 2 2 0）。そして、タスク B 用のメッセージ数カウンタの値をデクリメントする（S 2 3 0）。その結果、タスク B 用のメッセージ数カウンタの値は、0 となるので、このデータ受信利用処理を抜け（S 2 4 0：YES）、タスク B のその他の処理へ移行する。そして、タスク B のすべての処理が終了すると、処理を R T O S へ返す。

#### 【0 0 3 9】

従来は、図 5 に示したように、タスク B の起動終了が、送信したメッセージの数（3 回）分行われるため、非常にオーバーヘッドが大きかったのに対し、本実施例のタスク間通信方法では、図 3 に示したように、タスク B の起動終了は 1 回のみであり、タスクの起床・終了処理にかかるオーバーヘッドが少なくなる。そのため、C P U 1 1 の処理負荷を軽減することができる。

#### 【0 0 4 0】

特に、本実施例のようなエンジン制御では、例えば、急激に要求トルクが増加したことをセンサ 3 0 等から入力された信号に基づいて検知し燃料噴射を追加する為の非同期噴射要求、センサ 3 0 等の信号などの異常を検出しダイアグ用メモリに記憶させる為のダイアグ処理要求、外部ノードからの（CAN などによる）車内 L A N 3 2 を介した通信データの着信に対応する通信処理要求などを、メッセージとしてタスク間で通信する。こうした要求を検出する処理（送信側、すなわちタスク A）とは、別の優先度で、この対応をする処理（受信側、すなわち、タスク B）を実行したい場合や、複数の異なる優先度での送信処理に対し一つの受信処理を行う場合の受信処理側リソースの排他処理を簡単にする為に、上述したメッセージとしてタスク間通信でデータをやりとりする。

#### 【0 0 4 1】

これらのメッセージは、それぞれ要求が検出されるたびに送信され、以下のような場合に複数の受信側タスクが待ち状態になる。

- ・ 複数の送信処理が1つの高い優先度のタスク上で順次処理され、それぞれの送信条件が成立した場合。（例えば、4 m s 毎に非同期噴射判定と異常検出処理が行われている場合）

- ・ 優先度の高い複数のタスクが連続して実行されている時に、それぞれのタスク上の送信処理の送信条件が成立した場合。（例えば、CAN受信割り込みと 4 m s 毎の非同期噴射判定が重なった場合）

- ・ 一つの送信処理から（一つの送信条件により）、連続して複数のメッセージが送信された場合。（例えば、4 m s 毎の異常検出処理で、複数のセンサの異常が検出された場合）

このような場合に、上述した処理によって、CPU 11の処理負荷を軽減することができるため、エンジンの高回転時の制御性の悪化を防ぐことができる。

#### 【0042】

なお、本実施例では、タスクA内の処理からタスクB内の処理へデータを送るものとして説明したが、例えば、タスクA内の処理で送り先のタスクを指定するようにしてもよい。例えば、SendMessage関数の引数として、送り先のタスクを指定するようにしてもよい。送り先のタスクの指定は、例えば、タスクIDを引数とすることで行うことができる。タスクIDで行う場合には、図2（a）に示したS110～140の処理において、「タスクB用のキュー」という部分を、「引数として受け取ったタスクIDに対応するタスク用のキュー」とすることで実現できる。また、例えば、すべてのタスクの優先度が異なるものとして一意に決定されている場合には、送り先のタスクの指定は、この優先度を引数とすることで行うことができる。この場合、タスク別ではなく優先度別にキューを設け、図2（a）に示したS110～140の処理において、「タスクB用のキュー」という部分を、「引数として受け取った優先度に対応するキュー」とすることで実現できる。すなわち、メッセージ送信時に送信データのキューイングを行った後、該当する優先度のメッセージ送信カウンタの値に基づき、現在送信待ちの同優先度のメッセージが有るか否かを確認する。待ちメッセージが無い場合は従来

方式と同様にタスクの起床要求を R T O S に対して行うが、待ちメッセージがある場合はタスクの起床要求は行わず送信処理を終了する。そして R T O S によって受信側タスクが起床されると、このキューイングしたデータを取り出して、取り出したデータを利用する処理を行った後、さらに同優先度の送信待ちメッセージが残っているかどうかをチェックし、もし残っていればタスクを終了せずに続けて次の受信処理を実施し、送信待ちメッセージがなくなるまで繰り返す。このように構成することで、同一優先度のメッセージが複数連続して送信要求され、かつ、受信側タスクが送信側タスクに比べ優先度が同じか低い場合に、タスクの起床及び終了処理が少なく済み、C P U 1 1 の処理負荷が低減できる。

#### 【0043】

また、本実施例では、最初のメッセージであるか否かをメッセージ数カウンタの値に基づいて判定することにしたが、例えば、R T O S における起床待ちキューに送信先のタスクのタスク I D があるか否かによって判定するようにしてもよい。すなわち、図 2 の S 1 2 0、S 2 3 0 の処理を行わず、図 2 の S 1 3 0 の処理に代えて、「R T O S の起床待ちキューにタスク B のタスク I D がないか」を判定し、ない場合（S 1 3 0：YES）、S 1 4 0 の処理を行い、ある場合（S 1 4 0：NO）、S 1 4 0 の処理を行わずに終了するようにする。また、S 2 4 0 の処理に代えて、タスク B 用のキューにデータが残っていないかを判定し、残っている場合（S 2 4 0：NO）S 2 1 0 へ移行し、残っていない場合（S 2 4 0：NO）、データ受信利用処理を抜けるように構成すればよい。

#### 【図面の簡単な説明】

【図 1】 実施例の処理実行装置としてのエンジン制御装置の構成を表すブロック図である。

【図 2】 実施例のメッセージ送信処理とデータ受信処理の流れを示すフローチャートである。

【図 3】 実施例のタスク間通信方法によって、タスク A からタスク B に対してメッセージを 3 回発行した場合の処理の様子を示す説明図である。

【図 4】 従来のメッセージ送信処理とデータ受信処理の流れを示すフローチャートである。

【図 5】従来のタスク間通信方法によって、タスク A からタスク B に対してメッセージを 3 回発行した場合の処理の様子を示す説明図である。

【符号の説明】

1 … E C U

1 0 … マイコン

1 1 … C P U

1 2 … R O M

1 3 … R A M

1 4 … I / O

2 1 … 入力回路

2 2 … 出力回路

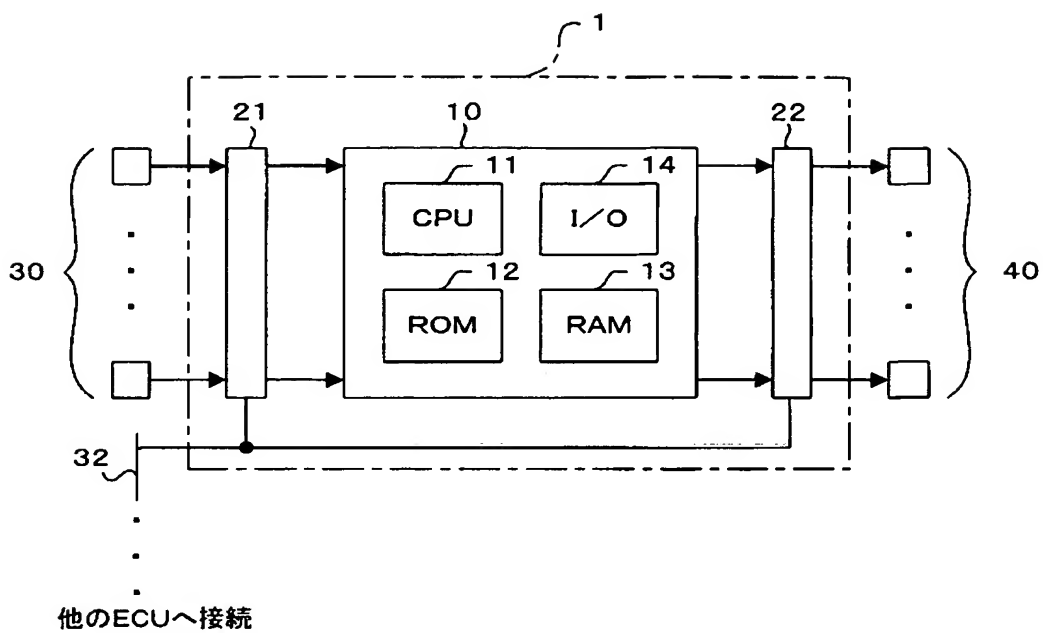
3 0 … センサ

3 2 … 車内 L A N

4 0 … アクチュエータ

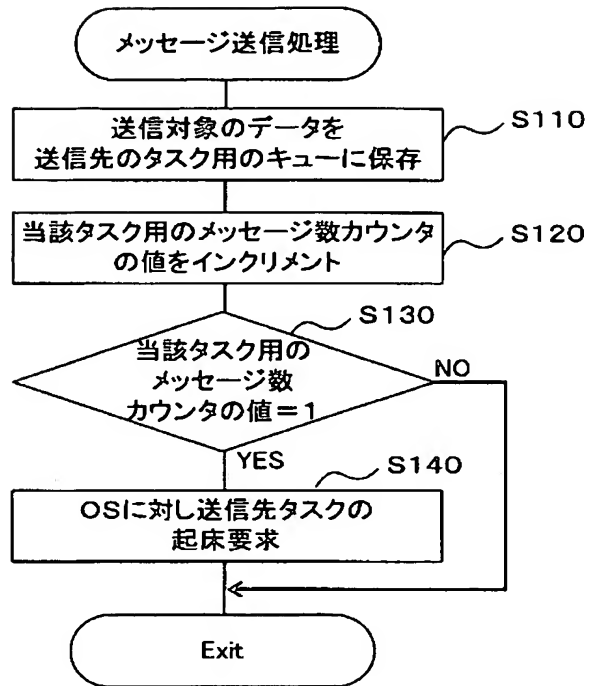
【書類名】 図面

【図 1】

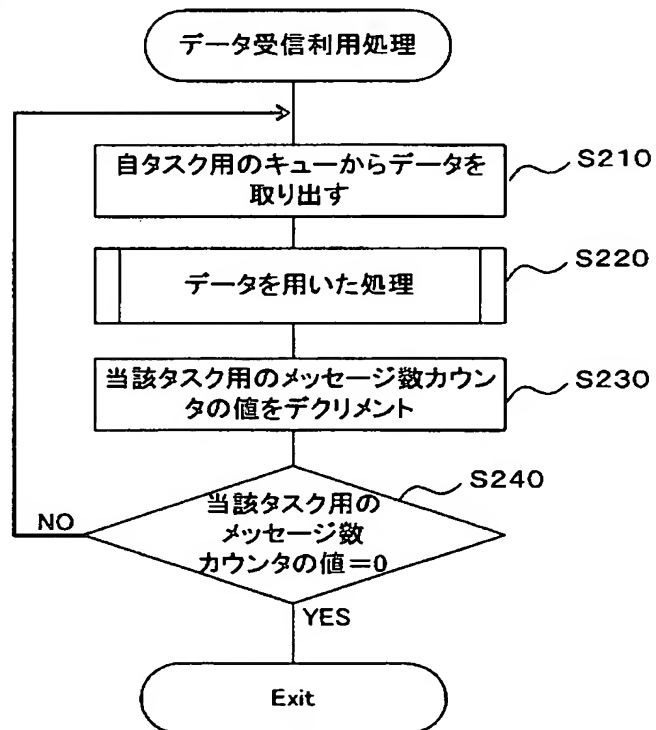


【図 2】

(a)

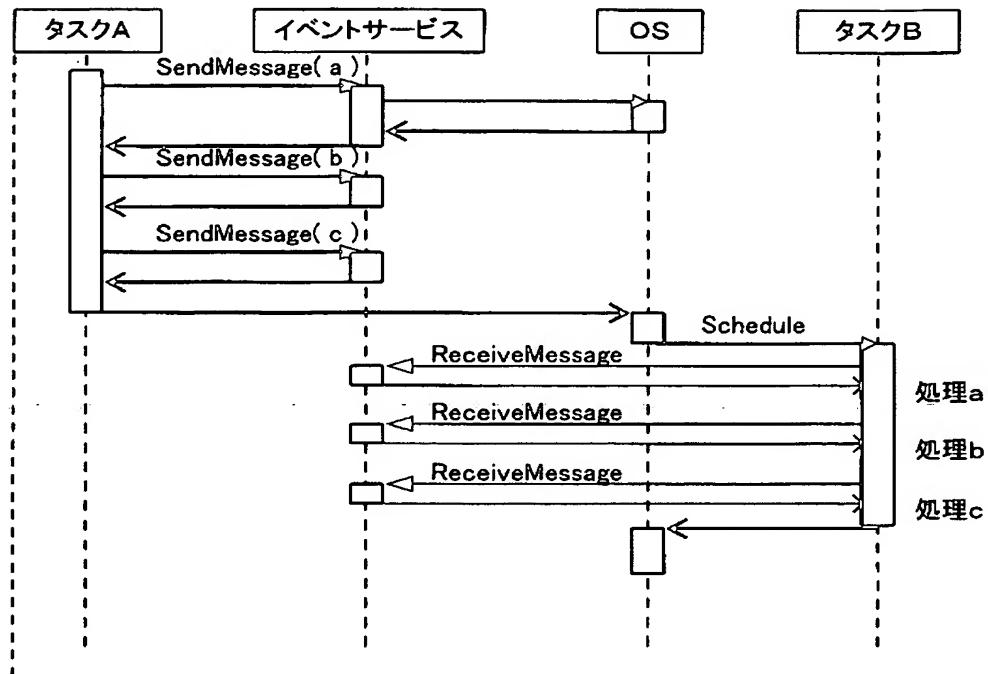


(b)



【図3】

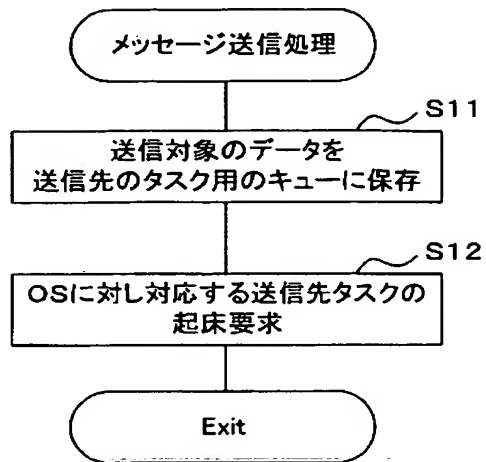
タスクAからタスクBへのメッセージを3回連続して発行した場合の例



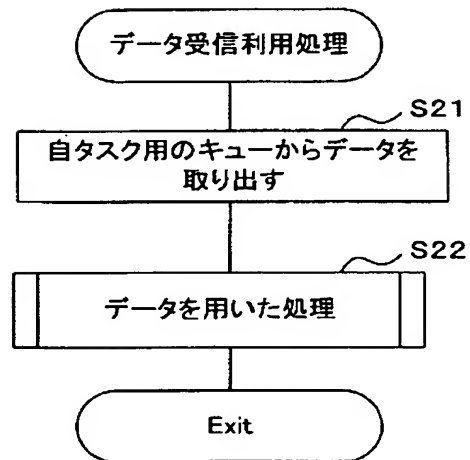


【図 4】

(a)



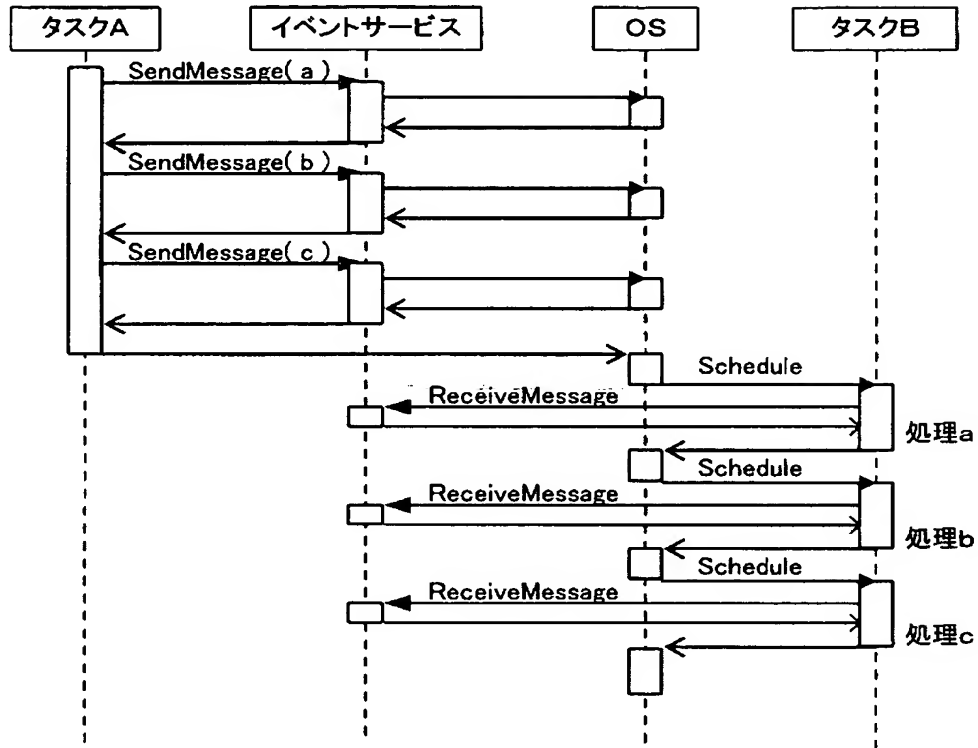
(b)



【図 5】

タスクAからタスクBへのメッセージを3回連続して発行した場合の例

・従来方式のシーケンス



**【書類名】 要約書****【要約】**

**【課題】** 処理装置の処理負荷を低減できるタスク間通信方法等を提供する。

**【解決手段】** メッセージ送信時にデータのキューイングを行った後、該当する優先度のメッセージ送信カウンタの値に基づき、現在送信待ちの同優先度のメッセージが有るか否かを確認する。待ちメッセージが無い場合は従来方式と同様にタスク B の起床要求を R T O S に対して行うが、待ちメッセージがある場合はタスク B の起床要求は行わず送信処理を終了する。そしてタスク B が起床されると、キューイングしたデータを取り出してそのデータを利用する処理を行った後、さらに同優先度の送信待ちメッセージが残っているかどうかをチェックし、もし残っていればタスクを終了せずに続けて次の受信処理を実施し、送信待ちメッセージがなくなるまで繰り返す。このように構成することで、タスクの起床及び終了処理が少なく済み、処理負荷を低減できる。

**【選択図】** 図 3

特願 2 0 0 3 - 0 4 0 0 5 6

出 願 人 履 歴 情 報

識別番号

[ 0 0 0 0 0 4 2 6 0 ]

1. 変更年月日

1 9 9 6 年 1 0 月 8 日

[変更理由]

名称変更

住 所

愛知県刈谷市昭和町 1 丁目 1 番地

氏 名

株式会社デンソー